L7: Entry 3 of 8 File: USPT May 30, 2006

DOCUMENT-IDENTIFIER: US 7055169 B2
TITLE: Supporting common interactive television functionality through presentation engine syntax

PRIOR-PUBLICATION:

| DOC-ID | DATE |
|---|---|
| US 20040139480 A1 | July 15, 2004 |

Brief Summary Text (7):
Interactive content such as application code or information relating to television programs may be broadcast in a cyclical or repeating format. The pieces of information which are broadcast in this manner form what may be referred to as a "carousel." A carousel may include multiple modules of data, including a directory module which indicates the particular modules which correspond to a given application. Frequently, a single carousel is transported as a contiguous data stream. However, it is also possible to multiplex two or more carousels in a single data stream. As an alternative to using a carousel format, some systems may utilize a return path to request and/or receive interactive content.

Brief Summary Text (8):
Broadcast systems may transmit information in a carousel format in order to allow receivers in the system to selectively obtain particular pieces of information in the carousel without requiring a return path from the receivers to the server. If a particular receiver needs a particular piece of information, it can simply wait until the next time that piece of information is broadcast, and then extract the information from the broadcast data stream. By employing carousels to broadcast information, the system may eliminate the need to connect each of the receivers with a server and further eliminate the need for the server to process individual requests for information.

Brief Summary Text (9):
The pieces of information, or data objects, in a carousel may be intended to be combined in a single object data stream to form a program. This program may also contain streaming data such as audio or video. For example, an interactive television game show may combine television audio and video with interactive content such as application code which allows users to answer questions. Another example would be a news program which combines audio and video with application code that inserts current stock prices in a banner at the bottom of the screen. Typically, each program is associated with a corresponding channel and, when a channel containing a particular program is selected by the interactive television receiver, the data which is being broadcast on that channel is downloaded and the program is started.

Description Paragraph (33):
First, HTML/JS content may be more dynamic than persistent. For example, in a current interactive television environment, an application may be configured to only execute code or use data that is packaged within the carousel in the same directory as the first program module. Hence, for security reasons the directory contents may clearly define the application boundary and the permissions signaled

within the directory may be applied to the entire contents of the directory. However, HTML/JS content may refer to other content (e.g., via a link) that is to be acquired from some location other than the carousel and the content that is referred to may replace the initial content. It is not clear that it is safe in this case to apply the same security permissions to such replacing content. Therefore, due to this dynamic nature, it is more difficult to define an "application boundary."

Description Paragraph (34):
Second, even when a product does not support multiple concurrent applications and restricts the application to only that content carried within the same directory in the carousel, there may be life cycle issues that affect the way that a content author designs the HTML/JS content. For example, if it is determined that the broadcaster can signal that an application may quit, it may be useful to invoke a handler written by the content author to respond to such an event. Similarly, there may be other states which might best be handled by an application-specific handler. For example, if the viewer is in the middle of a transaction involving an application, that application may wish to delay its termination until the transaction completes. Therefore, an application may be notified by the system when a broadcaster signals a new application available in the broadcast. In one embodiment, the application may be notified via an event, such as the O_exit event identified below. An application that determines that it does not want to exit immediately may extend its life by calling a defined event function such preventDefault( ).

Description Paragraph (55):
Identifies the currently tuned service_address and component_list for the primary pipe (see JS Tuning and Stream Selection below). This usage is similar to "tv:" in the DASE specification and "dvb://current.av" in the MHP specification. So, for example, this may be used within an HTML element to re-size and re-locate the currently playing video. broadcast://cnn.com Identifies the CNN TV channel and all of its component streams. This form of the URL can be used to request that the TV tuner switch channels. This URL in a service selection context causes the automatic selection of the default streams. That is, when used in a service selection context, the user-agent will (if the application is appropriately authorized) tune to the new channel and automatically select the default video stream, select the default audio stream (based on the preferred language), select the default sub-titles and teletext if identified in the user's current preferences, and select the default data carousel. broadcast://cnn.com;audio=eng Identifies the CNN TV channel and explicitly selects only the English audio stream. Documents use this form of the URL to explicitly reference a specific elementary stream. broadcast://current;audio=eng,video=current Selects the English audio stream on the current service. This URL allows the author to switch the current audio stream without explicitly knowing the current service address, and without changing the currently selected video stream. 2.1 HTML Tuning and Stream Selection

Description Paragraph (118):
In certain circumstances the specified rendering policy may not be possible, i.e., if a prerequisite resource has been removed from the carousel and acquisition via a modem has been denied by the viewer. In one embodiment, if no timeout for this loading has been specified, then the timeout may default to an indicated value (15s) as shown in the render-timeout property below. If a timeout occurs, and at least all of the prerequisite resources have been acquired, what is available for the new page may be displayed, independent of the specified rendering policy. If some of the prerequisite resources have not been acquired, then it may be preferable, if possible, for the display to show the previous page, if any. If this is not possible, then either an error message may appear or the box may render and display those resources which it has been able to acquire.

Description Paragraph (142):

The presence of path_segments in a URL indicates that it references a specific module in the data carousel associated with the service_address. For example, the URL "broadcast://tfl.fr/background.png" refers to the background.png module on the default data carousel.

Description Paragraph (145):
Load the module background.png from the default data carousel on the current service. broadcast://current;data=htp0/ Select the data carousel with track_tag "htp0", examine the directory module and load the "default" module in that directory (e.g., index.htm). Some applications may require the ability to load a specific module within a data carousel. For example, the following HTML loads the background.png module from the default carousel and uses it as a background image. <BODY background="broadcast:/background.png"> During a carousel request, typical HTML events which may be dispatched include. Load If the request succeeded, a load event is dispatched after the URL is finished loading. Error If the request is denied or otherwise invalid an error event is dispatched. Abort If the user aborts the request before it complete, an abort event is dispatched. Resident applications (such as a control task, or EPG) may require the ability to automatically launch an application during service selection. In these instances a URL of the form broadcast://cnn.com; data--htp0/ informs a browser to automatically execute the default module on a specific data carousel. Note: The default module may be selected by checking the specified directory for the following modules. The first module name that exists is automatically loaded. BHTTP Index.htp Index.htm A simpler URL of the form "broadcast:/" informs the browser to automatically execute the default module in the default carousel of the currently selected service. 4.1.2 Access Via the Http: URL Scheme and the Broadcast Carousel

Description Paragraph (146):
In one embodiment, HTML pages may use "http:" URLs to load resources from the carousel. In particular, the HTTP cache may be enhanced to automatically cache HTTP entities from the data carousel. Therefore, the http: URL handler will be able to load HTTP entities directly from the HTTP cache without opening an HTTP connection to the origin server. Hence, HTML pages that use an "http:" URL to reference HTTP entities may not notice any difference between resources retrieved from the broadcast and those retrieved using the client/server HTTP URL protocol.

Description Paragraph (147):
One embodiment of such a model is illustrated in FIG. 4. In the example of FIG. 4, the Head End 402 is acting as a proxy, is responsible for fetching data from the Origin Server 410 which has been requested by the carousel manager 420 (through as many hops as needed), and for placing the proper cache headers according to HTTP syntax and semantics (based upon expires header). The set-top-box 404 may then populate its cache from the carousel. The Expires entity-header field may gives the date/time after which the response is considered stale.

Description Paragraph (148):
In response to detecting an http url, the client-side may first check its local cache. If the requested data is not found in the cache, the client may check the current carousel if any, possibly retrieving data from the carousel. Alternatively, it may send an HTTP request for the specified URL.

Description Paragraph (149):
In order to allow proper cache behavior, the carousel may provide expiration dates and other cache control information in HTTP headers. For example, such information may include:

Description Paragraph (151):
It is noted that since network congestion can delay a response, revalidation of data which becomes obsolete during transit could result in an infinite loop. Consequently, HTTP 1.1 specifies that a response may not be revalidated in order to

avoid infinite loops. This rule may be followed whether the data comes from the carousel or directly from the origin server.

Description Paragraph (167):
Parameters typeArg of type DOMString Specifies the event type. canBubbleArg of type boolean Specifies whether or not the event can bubble. cancelableArg of type boolean Specifies whether or not the event's default action can be prevented. urlArg of type DOMString Specifies the Event's url. The different types of UrlEvents that can occur are: URLInserted The URLInserted event occurs when a URL is added to the carousel. Bubbles: Yes, Cancelable:No, Context Info: url URLUpdated The URLUpdated event occurs when a new version of an URL is created on the carousel. Bubbles: Yes, Cancelable: Yes, Context Info: url URLRemoved The URLRemoved event occurs when a URL is removed from the data carousel. Bubbles: Yes, Cancelable: No, Context Info: url The default action in the case of URLUpdated (which can be cancelled by calling preventDefault( )) is to reload the content of the associated url). There is no default action for URLInserted or URLRemoved. Also, note, that it is guaranteed that the events will be delivered in a top-down order; hence, if the body changes, then the event representing the update of the url associated with the body will be delivered prior to delivering any events concerning urls referred to by the body. Note that the above event can be signaled in the carousel by carrying a delta directory that indicates differences between the last directory and the current directory. That way, an implementation need not download the entire content before it knows whether the app is going to use it or not--it need only find out that there's a new version available. The Cache Object Introduction to Dynamic Cache Hints Since typically the amount of information that can be presented on a television screen is substantially less than contained in a page that is typically viewed on a PC, an author creating content for television will most often spread the same amount of information over multiple pages. Hence, the viewer will typically "scroll" between pages, and their navigation through a page can be a good indicator of which resources will be needed next. An author making use of such information by conveying hints based upon this navigation to the user agent can enable much better performance on lower end clients. A Host Cache Interface The cache interface supports two methods, prefetch and remove. The prefetch method specifies both the URL associated with the resource to be prefetched as well as a priority indicating how likely it is that the viewer will need that resource. The cache priority value is a non-negative integer. The author can use a cache priority value of 0 to indicate that the referenced content is useful, but that the author may be unsure of its likelihood of use in comparison with other items that they are requesting to be cached. The author can use a cache priority value of 1 to indicate the belief that caching the specified resource is very important. A very large value for the priority indicates that a resource will likely not be used (hence informing the user agent that it may reclaim the memory currently used to hold the URL's associated resource in cases where it is needed). The remove method may be used to remove a cached copy of the resource associated with the URL argument. Since it is to be removed from the cache, and not just invalidated, the system will not waste resources re-validating the entry. Note that invoking the remove method is different from assigning a very large integer as the cache priority value in that assigning such a large integer value only makes the space used to store that resource more available for garbage collecting and/or to hold high priority resources. Interface cache { void prefetch(in DOMString URL, in short priority); void remove (in DOMString URL); }; Binding of the Cache Interface to Script The Cache Object, which implements the cache interface above, is accessible as a property of the Navigator (Navigator::cache).

Description Paragraph (265):
In one embodiment, a directory module includes a corresponding per-application set of privileges that are requested. This directory module must contain a request for this set of privileges along with the producer's certificate and must be signed with the producer's private key. The producer's certificate is signed using the network's private key. The producer's certificate states the maximum privileges

that may be granted to any application under that producer. Hence, an application will only be granted a privilege if it is in its per-application set of privileges and it is among the set of maximum privileges that may be granted to any application associated with that producer. In addition to the signature, security is enhanced by requiring the signed directory to contain an accurate hash value corresponding to at least the initial code segment, and optionally to other code and data segments used by the application.

Previous Doc Next Doc Go to Doc#

L7: Entry 4 of 8 File: USPT May 17, 2005

DOCUMENT-IDENTIFIER: US 6895595 B2
TITLE: Module manager for interactive television system

Brief Summary Text (8):
Interactive television applications may consist of a set of program modules. The set of modules forming an application is typically self-contained in that all of the code needed by the application is in the set of modules. The first module is a directory module which identifies all of the modules which are part of the application. The entire set of modules, which is listed in the directory module, is transmitted via the broadcast channel to the set-top box and the application is executed. If a first interactive television application has completed execution and a second is to be executed, the directory and other modules of the second application are transmitted to the set-top box and the second application is executed. The entire set of modules used by the second application are transmitted even though some of the modules might be identical to modules used by the first application..

Brief Summary Text (10):
One of the advantages of designing software applications in a modular fashion is the ability to share modules between applications. The advantages of modularity may include conserving the limited amount of memory in a set-top box which can be used for interactive applications, reducing the time required to download applications from a broadcast station to a set-top box and reducing the amount of application code which must be written by allowing modules to be shared. The components of an application, however, may reside in different carousels (sets of modules, as described in more detail below) or in modules which may not be available at the time the application begins execution. It is therefore desirable to implement a system for management of the different modules. It would be advantageous for this management system to be capable of handling modules which have not yet been received and which may have to be extracted from an interactive television signal. Further, it would be advantageous for this management system to be capable of simultaneously monitoring several sources (e.g., a broadcast channel and a modem channel) for the modules needed by an application.

Brief Summary Text (11):
The invention comprises a system and method for managing modules of interactive television applications. One embodiment of the invention includes a set-top box configured with several input ports for receiving one or more interactive television signals which may embody the carousel modules. The interactive television signals typically comprise packets of compressed data corresponding to modules of an interactive television application, television programs, or other interactive television data. ("Television program" as used herein refers generally to any type of audio and/or video programming which is normally viewed on a television.) The various types of packets are generally time multiplexed with each other. The set-top box monitors the input ports to determine whether a packet received at the input port contains module data and, if so, whether the module should be processed and stored in the set-top box. If the system determines that the module should be stored, the packets corresponding to the module are extracted from the interactive television signal. When all of the packets corresponding to the module have been extracted from the interactive television signal, the module

is processed and made available for execution.

Brief Summary Text (12):
If there is no interactive television application executing in the set-top box, the system monitors the input ports for auto-loading modules, such as application directory modules. When one of these application directory modules is detected, the system begins collecting packets containing data from the directory module. When all of these packets have been collected, the system reconstructs the directory module and authenticates it. The system then monitors the input ports for the carousel modules (i.e., those listed in the directory module,) stores the corresponding packets and reconstructs the modules for use by the application corresponding to the carousel.

Drawing Description Text (5):
FIG. 3 is an illustration of the component modules of a carousel and the transmission order of the modules in one embodiment of the invention.

Detailed Description Text (3):
One embodiment of the invention is described below. In this embodiment, an interactive television receiver accepts an audio-video-interactive signal via a broadcast channel such as direct satellite transmission. ("Direct" satellite transmission as used herein contemplates transmissions received by the interactive television receiver, more particularly by its antenna, directly from the satellite.) The audio-video-interactive signal contains television programs or similar audio-video content, as well as interactive content such as control signals or interactive applications. ("Broadcast" is used herein to refer to transmission of a single signal to all subscribing receivers.) The interactive television receiver is also configured to receive signals via a modem channel. The signals transmitted via the broadcast and modem channels may both embody various modules. Each of these two channels is therefore considered a "module source" for the purposes of this description. The modules contained in the signals from the module sources may comprise components of an interactive application. The modules can contain any type of data, such as application code, raw data or graphical information. Because the modules are generally transmitted to the television receiver in a cyclic manner, the set of modules is often referred to as a carousel. (An application is a carousel which contains a "top-level" program.) A module manager is implemented in the system to control the manner in which the modules are requested by an application, received from the module sources and matched with the application requiring the modules.

Detailed Description Text (4):
If no application is executing in the system, the module manager monitors the module sources for modules which are auto-loading. The directory module of an interactive application, for example, may be auto-loading. If an application is executing in the system, the application may request modules. More particularly, it may request modules which are not part of the same carousel as the application. The module manager stores the application's requests and monitors the module sources for modules corresponding to the requests. When one of the requested modules is detected, the module manager stores it, makes it available to the executing application and removes the request from the module manager's request queue. If the module manager receives a module which has not been requested and which cannot be automatically loaded, the module is ignored (i.e., it is not stored in the system.)

Detailed Description Text (12):
Referring to FIG. 3, an application (and in fact, any carousel) consists of a series of modules, one of which is a directory module. The directory module has a unique identifier so that it can be identified during transmission without further information. The directory module contains an entry for each of the modules in the application and any module which does not have a corresponding entry in the

directory module is not recognized by the application. The directory module contains enough information to allow the interactive television receiver to access all of the parts (i.e., modules) of the application which may be necessary for execution of the program. The directory module must be accessed before the other modules of the application so that the remainder of the modules can be properly interpreted. The directory module may be transmitted several times during the cycle in which the modules of the application are transmitted in order to ensure that it is available for essentially random access to the other modules.

Detailed Description Text (13):
The directory modules of all the applications have a common format. The format consists of three parts: a portion having fixed-length components; a portion having variable-length components; and a portion having certification information. The fixed-length portion contains data on the application and each of the modules in the application. The variable-length portion contains string data on the module names and the hash of the modules. The certification portion contains the producer certificate and directory signature.

Detailed Description Text (15):
Referring to FIG. 4, each of the modules 51 has a data segment 52 and a CRC segment 53. The data segment 52 of the directory module is described above. The data segment 52 of the remainder of the modules can contain any type of data, such as application code or raw data. The CRC segment 53 of each of the modules is used for error control and is computed for the entire module 51. Each of the modules 51 has a unique identifier.

Detailed Description Text (19):
Referring to FIG. 5, a diagrammatic representation of the signal transmitted from broadcast station 10 to receiving station 20 is shown. The packets of several program sources are multiplexed into a single transmission stream if necessary. These packets may contain data for various applications or television programs. The illustrated transmission stream includes audio (A) and video (V) packets of a television program, as well as packets of two interactive application modules (M1, M2.) The packets are formatted as explained above to enable reconstruction of the packets into the respective programs and modules. It should be noted that several modules can be simultaneously transmitted by combining their packets in the transmission signal. The figure illustrates the time multiplexing of the packets of the modules and the television program. The modules need not belong to the same carousel to be transmitted together. It can be seen from the figure that there are typically more packets of video data for a particular television program than audio data for that program as a result of the greater amount of video data which typically must be transmitted.

Detailed Description Text (20):
The broadcast signal is received by set-top box 22, which demultiplexes the packets and reconstructs the respective television programs and modules. As explained above, the modules are stored in RAM 37, where they are available for use by applications executing in the control system 35. The set-top box may employ a security mechanism to ensure that the carousels and/or particular modules which are being downloaded are authentic. A certificate system may be used to ensure that the modules are produced by authorized producers and that the modules have not been altered before being received by the set-top box.

Detailed Description Text (22):
The set-top box maintains copies of the public keys of one or more trusted parties. When the set-top box receives a directory module, it checks the module for a certificate signed with the private key of the producer. The certificate contains a producer's certificate, which is the producer's public key, signed by a trusted party. The set-top box, having a copy of the trusted party's public key, can verify that the producer's certificate (the producer's public key) is authentic. Then, the

producer's authenticated public key can be used to verify that the certificate is unaltered. The security mechanism may also include performing a hash function over the modules and including the hash value in the corresponding directory module or the like. Credentialing or other security in the directory module is typically implemented after the insertion of the other modules' hash values with their entries in the directory module.

Detailed Description Text (23):
The module manager controls delivery, access and other functions which relate to the modules. Referring to FIG. 6, a block diagram illustrating the organization of one embodiment of the module manager 60 is shown. Module manager 60 includes a series of interfaces 61-63, a series of databases 64-66 and a post-load manager 67. The interfaces are source interface 61, client interface 62 and control task interface 63. The databases include, credential database 64, carousel database 65 and request database 66.

Detailed Description Text (26):
Credential database 64 contains information on the producers of various carousels and the credentials associated with the producers. Credential database 64 is implemented as a linked list of producer entries, each entry having an ID for a given producer, a pointer to the next producer entry and a pointer to a first credential in a linked list of credentials for the given producer. The credentials in the linked list for each producer are verified before the credentials are entered in the database, so the signature and credential flags are removed before the credentials are stored. Each credential entry includes a pointer to the next credential entry, IDs for the carousels granting and being granted access, an expiration date and various flags. Accesses to credential database 64 may include adding, removing or finding specific entries, and removing groups of entries which are old or non-global.

Detailed Description Text (27):
Carousel database 65 contains information regarding carousels of which module manager 60 is aware. Carousel database 65 is implemented as a linked list of carousel entries. Each entry includes a carousel ID, various flags and pointers and a reference counter. The pointers indicate the locations of the directory module for the corresponding carousel and a list of associated requests in the request database. Carousel database 65 can be accessed by finding, adding or deleting entries, and by reading or manipulating data within the entries. Some of the carousel database access routines also manipulate things other than carousel database 65. For example, if the buffer in which the directory module should be loaded is requested, but no such buffer is assigned, a temporary buffer will be allocated.

Detailed Description Text (28):
Request database 66 contains information on pending requests for various modules. Request database 16 is implemented as a set of linked lists, each of which is associated with a corresponding carousel. As pointed out above, each entry in carousel database 65 has a pointer to a corresponding linked list of associated requests in request database 66. Each entry is also recorded in a hash table in order to reduce the amount of time which may be required to search for particular entries. Each entry in request database 66 has a pointer to the next entry, IDs for the requested module and its carousel, various flags and state and buffer information. Each entry stores a pointer to a buffer in which the requested module is to be stored, as well as the size of the buffer. Request database 66 may include a temporary storage allocation component which allows the request database to receive modules which have not been requested or for which buffers have not been allocated. Because the temporary storage allocation component of the database may not be optimized to efficiently store these modules, they should be moved to more permanent storage as soon as possible. Request database 66 implements functions to find, add, delete or read entries as well as to update the state of the requests as

described below.

Detailed Description Text (32):
When a module is detected, the module management unit begins loading the module into memory. Once a module is completely loaded, it is entered in a list of stored modules. It is then tested to determine whether it is a directory module. If the module is a directory module, the module management unit authenticates the module's producer certificate. The module management unit may be programmed to take any of several actions if the certificate is not authentic, such as simply discarding the module, warning the user, or shutting down the set-top box. If the certificate is authentic, the module may be checked to verify its integrity according to the security mechanisms implemented in the system. If it is determined that the module contains errors, it is discarded. If the module is error-free, it is made available to the system for use in downloading other modules and providing information about the modules for execution purposes.

Detailed Description Text (33):
After start up, the module management unit may download modules in response to requests or as a result of being listed in the directory module (collectively, "needed" modules.) Requests for particular modules are placed in a list of pending requests maintained by the module management unit. The module management unit waits, monitoring the module sources for packets corresponding to the needed modules. The packets are detected and separated from the incoming stream of packets in the manner described above for the directory module. The module management unit may retrieve packets of more than one module at a time if they are interspersed. When receipt of all the packets for a module is finished, the module is reconstructed. The module may be tested for errors and its authenticity may be verified if the appropriate error-checking and security mechanisms have been implemented. If the completed module corresponds to a pending request, the module is matched to that request (i.e., the request is canceled and the module is made available to the application or module which initiated the request.) When the module management unit completes downloading of the needed modules which were available at the module sources, it resumes waiting for the next needed module.

CLAIMS:

2. The receiver as recited in claim 1, wherein the interactive television application comprises a plurality of interactive television application modules corresponding to a first carousel, and wherein said requested interactive television application modules are part of a second carousel.

3. The receiver as recited in claim 2, wherein said microprocessor is further configured to: execute a second application comprising a plurality of modules corresponding to a third carousel; and store one or more requests generated by said second application for said requested modules.

4. The receiver as recited in claim 1, wherein said microprocessor is further configured to retrieve modules from one or both of said broadcast signal and said second signal in response to detecting said modules are listed in a directory module.

6. A method for managing carousel modules in an interactive television system, the method comprising: executing an interactive television application; storing in a data storage device one or more requests generated by said interactive television application for one or more corresponding requested interactive television application modules; monitoring a plurality of ports for said requested interactive television application modules; retrieving said requested interactive television application modules from one or more of said ports; and storing said retrieved interactive television application modules.

7. The method as recited in claim 6, wherein the interactive television application comprises a plurality of modules corresponding to a first carousel, and wherein said requested modules are part of a second carousel.

8. The method as recited in claim 7, further comprising: executing a second application comprising a plurality of modules corresponding to a third carousel; and storing one or more requests generated by said second application for said requested modules.

9. The method as recited in claim 6, further comprising retrieving modules front one or more of said ports in response to detecting said modules are listed in a directory module.

12. The system as recited in claim 11, wherein the interactive television application comprises a plurality of interactive television application modules corresponding to a first carousel, and wherein said requested interactive television application modules are part of a second carousel.

13. The system as recited in claim 12, wherein said receiver is further configured to store one or more requests generated by a second application for said requested modules, wherein said second application comprises a plurality of modules corresponding to a third carousel.

14. The system as recited in claim 11, wherein said receiver is further configured to retrieve modules from one or both of said broadcast signal and said second signal in response to detecting said modules are listed in a directory module.

17. The storage medium as recited in claim 16, wherein the interactive television application comprises a plurality of interactive television application modules corresponding to a first carousel, and wherein said requested interactive television application modules are part of a second carousel.

18. The storage medium as recited in claim 17, wherein said program instructions are executable to: execute a second application comprising a plurality of modules corresponding to a third carousel; and store one or more requests generated by said second application for said requested modules.

19. The storage medium as recited in claim 16, wherein said program instructions are executable to retrieve modules from one or both of said broadcast signal and said second signal in response to detecting said modules are listed in a directory module.

Previous Doc Next Doc Go to Doc#

L9: Entry 1 of 1 File: USPT May 30, 2006

DOCUMENT-IDENTIFIER: US 7055169 B2
TITLE: Supporting common interactive television functionality through presentation engine syntax

PRIOR-PUBLICATION:

| DOC-ID | DATE |
|---|---|
| US 20040139480 A1 | July 15, 2004 |

Brief Summary Text (7):
Interactive content such as application code or information relating to television programs may be broadcast in a cyclical or repeating format. The pieces of information which are broadcast in this manner form what may be referred to as a "carousel." A carousel may include multiple modules of data, including a directory module which indicates the particular modules which correspond to a given application. Frequently, a single carousel is transported as a contiguous data stream. However, it is also possible to multiplex two or more carousels in a single data stream. As an alternative to using a carousel format, some systems may utilize a return path to request and/or receive interactive content.

Brief Summary Text (8):
Broadcast systems may transmit information in a carousel format in order to allow receivers in the system to selectively obtain particular pieces of information in the carousel without requiring a return path from the receivers to the server. If a particular receiver needs a particular piece of information, it can simply wait until the next time that piece of information is broadcast, and then extract the information from the broadcast data stream. By employing carousels to broadcast information, the system may eliminate the need to connect each of the receivers with a server and further eliminate the need for the server to process individual requests for information.

Brief Summary Text (9):
The pieces of information, or data objects, in a carousel may be intended to be combined in a single object data stream to form a program. This program may also contain streaming data such as audio or video. For example, an interactive television game show may combine television audio and video with interactive content such as application code which allows users to answer questions. Another example would be a news program which combines audio and video with application code that inserts current stock prices in a banner at the bottom of the screen. Typically, each program is associated with a corresponding channel and, when a channel containing a particular program is selected by the interactive television receiver, the data which is being broadcast on that channel is downloaded and the program is started.

Description Paragraph (33):
First, HTML/JS content may be more dynamic than persistent. For example, in a current interactive television environment, an application may be configured to only execute code or use data that is packaged within the carousel in the same

directory as the first program module. Hence, for security reasons the directory contents may clearly define the application boundary and the permissions signaled within the directory may be applied to the entire contents of the directory. However, HTML/JS content may refer to other content (e.g., via a link) that is to be acquired from some location other than the carousel and the content that is referred to may replace the initial content. It is not clear that it is safe in this case to apply the same security permissions to such replacing content. Therefore, due to this dynamic nature, it is more difficult to define an "application boundary."

Description Paragraph (34):
Second, even when a product does not support multiple concurrent applications and restricts the application to only that content carried within the same directory in the carousel, there may be life cycle issues that affect the way that a content author designs the HTML/JS content. For example, if it is determined that the broadcaster can signal that an application may quit, it may be useful to invoke a handler written by the content author to respond to such an event. Similarly, there may be other states which might best be handled by an application-specific handler. For example, if the viewer is in the middle of a transaction involving an application, that application may wish to delay its termination until the transaction completes. Therefore, an application may be notified by the system when a broadcaster signals a new application available in the broadcast. In one embodiment, the application may be notified via an event, such as the O_exit event identified below. An application that determines that it does not want to exit immediately may extend its life by calling a defined event function such preventDefault( ).

Description Paragraph (55):
Identifies the currently tuned service_address and component_list for the primary pipe (see JS Tuning and Stream Selection below). This usage is similar to "tv:" in the DASE specification and "dvb://current.av" in the MHP specification. So, for example, this may be used within an HTML element to re-size and re-locate the currently playing video. broadcast://cnn.com Identifies the CNN TV channel and all of its component streams. This form of the URL can be used to request that the TV tuner switch channels. This URL in a service selection context causes the automatic selection of the default streams. That is, when used in a service selection context, the user-agent will (if the application is appropriately authorized) tune to the new channel and automatically select the default video stream, select the default audio stream (based on the preferred language), select the default sub-titles and teletext if identified in the user's current preferences, and select the default data carousel. broadcast://cnn.com;audio=eng Identifies the CNN TV channel and explicitly selects only the English audio stream. Documents use this form of the URL to explicitly reference a specific elementary stream. broadcast://current;audio=eng,video=current Selects the English audio stream on the current service. This URL allows the author to switch the current audio stream without explicitly knowing the current service address, and without changing the currently selected video stream. 2.1 HTML Tuning and Stream Selection

Description Paragraph (118):
In certain circumstances the specified rendering policy may not be possible, i.e., if a prerequisite resource has been removed from the carousel and acquisition via a modem has been denied by the viewer. In one embodiment, if no timeout for this loading has been specified, then the timeout may default to an indicated value (15s) as shown in the render-timeout property below. If a timeout occurs, and at least all of the prerequisite resources have been acquired, what is available for the new page may be displayed, independent of the specified rendering policy. If some of the prerequisite resources have not been acquired, then it may be preferable, if possible, for the display to show the previous page, if any. If this is not possible, then either an error message may appear or the box may render and display those resources which it has been able to acquire.

Description Paragraph (142):
The presence of path_segments in a URL indicates that it references a specific module in the data carousel associated with the service_address. For example, the URL "broadcast://tfl.fr/background.png" refers to the background.png module on the default data carousel.

Description Paragraph (145):
Load the module background.png from the default data carousel on the current service. broadcast://current;data=htp0/ Select the data carousel with track_tag "htp0", examine the directory module and load the "default" module in that directory (e.g., index.htm). Some applications may require the ability to load a specific module within a data carousel. For example, the following HTML loads the background.png module from the default carousel and uses it as a background image. <BODY background="broadcast:/background.png"> During a carousel request, typical HTML events which may be dispatched include. Load If the request succeeded, a load event is dispatched after the URL is finished loading. Error If the request is denied or otherwise invalid an error event is dispatched. Abort If the user aborts the request before it complete, an abort event is dispatched. Resident applications (such as a control task, or EPG) may require the ability to automatically launch an application during service selection. In these instances a URL of the form broadcast://cnn.com; data--htp0/ informs a browser to automatically execute the default module on a specific data carousel. Note: The default module may be selected by checking the specified directory for the following modules. The first module name that exists is automatically loaded. BHTTP Index.htp Index.htm A simpler URL of the form "broadcast:/" informs the browser to automatically execute the default module in the default carousel of the currently selected service. 4.1.2 Access Via the Http: URL Scheme and the Broadcast Carousel

Description Paragraph (146):
In one embodiment, HTML pages may use "http:" URLs to load resources from the carousel. In particular, the HTTP cache may be enhanced to automatically cache HTTP entities from the data carousel. Therefore, the http: URL handler will be able to load HTTP entities directly from the HTTP cache without opening an HTTP connection to the origin server. Hence, HTML pages that use an "http:" URL to reference HTTP entities may not notice any difference between resources retrieved from the broadcast and those retrieved using the client/server HTTP URL protocol.

Description Paragraph (147):
One embodiment of such a model is illustrated in FIG. 4. In the example of FIG. 4, the Head End 402 is acting as a proxy, is responsible for fetching data from the Origin Server 410 which has been requested by the carousel manager 420 (through as many hops as needed), and for placing the proper cache headers according to HTTP syntax and semantics (based upon expires header). The set-top-box 404 may then populate its cache from the carousel. The Expires entity-header field may gives the date/time after which the response is considered stale.

Description Paragraph (148):
In response to detecting an http url, the client-side may first check its local cache. If the requested data is not found in the cache, the client may check the current carousel if any, possibly retrieving data from the carousel. Alternatively, it may send an HTTP request for the specified URL.

Description Paragraph (149):
In order to allow proper cache behavior, the carousel may provide expiration dates and other cache control information in HTTP headers. For example, such information may include:

Description Paragraph (151):
It is noted that since network congestion can delay a response, revalidation of

data which becomes obsolete during transit could result in an infinite loop. Consequently, HTTP 1.1 specifies that a response may not be revalidated in order to avoid infinite loops. This rule may be followed whether the data comes from the carousel or directly from the origin server.

Description Paragraph (167):
Parameters typeArg of type DOMString Specifies the event type. canBubbleArg of type boolean Specifies whether or not the event can bubble. cancelableArg of type boolean Specifies whether or not the event's default action can be prevented. urlArg of type DOMString Specifies the Event's url. The different types of UrlEvents that can occur are: URLInserted The URLInserted event occurs when a URL is added to the carousel. Bubbles: Yes, Cancelable:No, Context Info: url URLUpdated The URLUpdated event occurs when a new version of an URL is created on the carousel. Bubbles: Yes, Cancelable: Yes, Context Info: url URLRemoved The URLRemoved event occurs when a URL is removed from the data carousel. Bubbles: Yes, Cancelable: No, Context Info: url The default action in the case of URLUpdated (which can be cancelled by calling preventDefault( )) is to reload the content of the associated url). There is no default action for URLInserted or URLRemoved. Also, note, that it is guaranteed that the events will be delivered in a top-down order; hence, if the body changes, then the event representing the update of the url associated with the body will be delivered prior to delivering any events concerning urls referred to by the body. Note that the above event can be signaled in the carousel by carrying a delta directory that indicates differences between the last directory and the current directory. That way, an implementation need not download the entire content before it knows whether the app is going to use it or not--it need only find out that there's a new version available. The Cache Object Introduction to Dynamic Cache Hints Since typically the amount of information that can be presented on a television screen is substantially less than contained in a page that is typically viewed on a PC, an author creating content for television will most often spread the same amount of information over multiple pages. Hence, the viewer will typically "scroll" between pages, and their navigation through a page can be a good indicator of which resources will be needed next. An author making use of such information by conveying hints based upon this navigation to the user agent can enable much better performance on lower end clients. A Host Cache Interface The cache interface supports two methods, prefetch and remove. The prefetch method specifies both the URL associated with the resource to be prefetched as well as a priority indicating how likely it is that the viewer will need that resource. The cache priority value is a non-negative integer. The author can use a cache priority value of 0 to indicate that the referenced content is useful, but that the author may be unsure of its likelihood of use in comparison with other items that they are requesting to be cached. The author can use a cache priority value of 1 to indicate the belief that caching the specified resource is very important. A very large value for the priority indicates that a resource will likely not be used (hence informing the user agent that it may reclaim the memory currently used to hold the URL's associated resource in cases where it is needed). The remove method may be used to remove a cached copy of the resource associated with the URL argument. Since it is to be removed from the cache, and not just invalidated, the system will not waste resources re-validating the entry. Note that invoking the remove method is different from assigning a very large integer as the cache priority value in that assigning such a large integer value only makes the space used to store that resource more available for garbage collecting and/or to hold high priority resources. Interface cache { void prefetch(in DOMString URL, in short priority); void remove (in DOMString URL); }; Binding of the Cache Interface to Script The Cache Object, which implements the cache interface above, is accessible as a property of the Navigator (Navigator::cache).

Description Paragraph (265):
In one embodiment, a directory module includes a corresponding per-application set of privileges that are requested. This directory module must contain a request for this set of privileges along with the producer's certificate and must be signed

with the producer's private key. The producer's certificate is signed using the network's private key. The producer's certificate states the maximum privileges that may be granted to any application under that producer. Hence, an application will only be granted a privilege if it is in its per-application set of privileges and it is among the set of maximum privileges that may be granted to any application associated with that producer. In addition to the signature, security is enhanced by requiring the signed directory to contain an accurate hash value corresponding to at least the initial code segment, and optionally to other code and data segments used by the application.

Other Reference Publication (1):
"CSS3 Module: The Box Model"; W3C Working Draft, Jul. 26, 2001; This version: http://www.w3.org/TR/2001/WD-css3-box-20010726; Latest version: http://www.w3.org/TR/css3-box; Editor: Bert Bos; Copyright .COPYRGT. 2001 W3C.RTM. (MIT, INRIA, Keio); pp. 1-104. cited by other

Previous Doc Next Doc Go to Doc#

# Freeform Search

**Database:**
US Pre-Grant Publication Full-Text Database
US Patents Full-Text Database
US OCR Full-Text Database
EPO Abstracts Database
JPO Abstracts Database
Derwent World Patents Index
IBM Technical Disclosure Bulletins

**Term:** 20020059645.PN.

**Display:** 50 **Documents in Display Format:** - **Starting with Number** 1

**Generate:** Hit List [✓ Hit Count] Side by Side Image

Search Clear Interrupt

## Search History

**DATE: Friday, January 04, 2008** Purge Queries Printable Copy Create Case

| **Set Name** side by side | **Query** | **Hit Count** | **Set Name** result set |
|---|---|---|---|
| *DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR* | | | |
| L12 | L1O AND CAROUSEL | 0 | L12 |
| *DB=USPT; PLUR=YES; OP=OR* | | | |
| L11 | 20020059645.PN. AND CAROUSEL | 0 | L11 |
| *DB=PGPB,USPT,USOC,EPAB,JPAB,DWPI,TDBD; PLUR=YES; OP=OR* | | | |
| L10 | 20020059645.PN. | 2 | L10 |
| *DB=USPT; PLUR=YES; OP=OR* | | | |
| L9 | l8 and (DELTA) | 1 | L9 |
| L8 | l7 and version | 4 | L8 |
| L7 | L6 and carousel | 8 | L7 |
| L6 | directory near modules | 216 | L6 |
| L5 | L4 and (main near directory) | 0 | L5 |
| L4 | delta near directory | 8 | L4 |
| L3 | L2 and (delta near directory) | 0 | L3 |
| L2 | main near directory | 437 | L2 |
| L1 | main near directory near module$ | 0 | L1 |

END OF SEARCH HISTORY

L7: Entry 2 of 8 File: USPT Jun 27, 2006

DOCUMENT-IDENTIFIER: US 7069572 B2
TITLE: Broadcast data access system for multimedia clients in a broadcast network architecture

PRIOR-PUBLICATION:

| DOC-ID | DATE |
|---|---|
| US 20030177230 A1 | September 18, 2003 |

Abstract Text (1):
A broadcast data access system is provided for receiving broadcast data by applications residing on a multimedia client, where the broadcast data is a set of modules on a data carousel that are being broadcast over a broadcast network. The broadcast data access system includes an interest manager configured to store a plurality of interests, such that each interest identifies an available module on the data carousel being requested by an application. The system further includes at least one application having registered an interest for a first module with the interest manager, and a dispatcher distributing the first module to the requesting application by accessing the interest manager.

Brief Summary Text (2):
The present invention relates generally to a broadcast data access system, and more particularly to an architecture for supporting applications that receive broadcast data from a data carousel over a broadcast network.

Brief Summary Text (4):
A broadcast data carousel is commonly used for transporting data in a broadcast environment. This underlying mechanism for transporting data is defined in the MPEG-2 DSM-CC specification (i.e., ISC/IEC 13818-6). Using this mechanism, the server repeatedly sends data over a period of time so that a client who is interested in the data may receive it only when it is required. If a client misses some of the data or receives defective data, it waits for the next broadcast of the data to receive any data that it may need.

Brief Summary Text (5):
A Broadcast File System (BFS) provides a layer on top of the broadcast data carousel that hides the details of the underlying transport mechanism from the server and clients. In particular, BFS creates a mapping between the carousel/file number and a module name. As a result, the server and clients view the broadcast data as a standard hierarchical file system similar to files found on a disk operating system.

Brief Summary Text (6):
Therefore, it would be desirable to provide a broadcast data access system for receiving broadcast data from a data carousel in a simple, efficient, and flexible manner. It should support multiple data sources between the broadcast network and the multimedia client, such that each source can receive a different form of encoded broadcast data. In addition, the broadcast data access system should be able to efficiently process data packets received in a non-sequential order, as well as simultaneously fulfill multiple requests for the same data packets by

different applications. To lessen processing overhead, filters are dynamically installed on the client. Lastly, the present invention should provide a method for downloading and synchronizing a directory module with the content of the data carousel being broadcast to the client.

Description Paragraph (10):
A Broadcast File System 10 (BFS), as depicted in FIG. 1, provides an architecture for delivering various types of data from a BFS server 12 (or group of servers) to a plurality of multimedia clients 14 over a broadcast network 16. It is also envisioned that a two-way communication network could be used in conjunction with the broadcast file system 10 of the present invention. The underlying mechanism for transporting data across the network 16 relies on a broadcast data carousel that is defined in the MPEG-2 DSM-CC specification (i.e., ISC/IEC 13818-6). Typically, broadcast data is grouped into files that are subdivided into fixed-size data blocks and then broadcast in a non-sequential order using the data carousel mechanism. However, BFS 10 of the present invention provides a layer on top of the broadcast data carousel that hides the details of this underlying transport mechanism from the server 12 and clients 14. Within a data carousel, individual data files are called modules. Since modules are identified by numbers (not names), BFS 10 creates a mapping between file numbers and module names. In this way, the server 12 and clients 14 of BFS 10 view these modules in a standard hierarchical file system similar to files found on a disk operating system.

Description Paragraph (11):
BFS server 12 is the component responsible for storing, assembling and delivering modules across the network 16. While the following discussion is provided with reference to one data carousel, it is readily understood that the explanation is applicable to more than one data carousel. A top level data carousel contains at least one module known as the BFS directory which includes the module names for all of the other modules on this or any other data carousel. As modules are added to the data carousel, BFS server 12 creates a module name (i.e., identifier) for each new module and then updates the BFS directory structure. Similarly, when modules are updated and/or deleted from the data carousel, the BFS directory structure is updated by BFS server 12. Applications residing on a multimedia client 14 in turn utilize the BFS directory to access modules contained on the broadcast data carousel. The network 16 may employ any underlying transport protocol (e.g., MPEG transport and/or UDP/IP) that has the ability to deliver data packets across the network to the client.

Description Paragraph (12):
A broadcast data access system 30 of the present invention serves as the interface between the set of modules contained on the data carousel 32 and the various applications residing 34 on multimedia client 14. Referring to FIG. 2, the broadcast data access system 30 on each multimedia client 14 includes an interest manager 36 that is configured to store a plurality of interests. To receive broadcast data from the data carousel 32, an application 34 registers an interest with the interest manager 36 (e.g., a first application registers a first interest requesting a first module from the data carousel). For purpose of this discussion, the term "application" signifies any software module, including the operating system, which may reside on the multimedia client 14. For each module received by the multimedia client 14, a dispatcher 38 accesses the interest manager 36 to determine if any application residing on that client 14 has requested that module, and if so distributes that module to the appropriate application 34.

Description Paragraph (14):
Once pre-processed by a data source processor 42, data blocks are passed along (in DSM-CC format) to a dispatcher 38. Dispatcher 38 may perform some additional data block verification (e.g., check payload length). However, based on the particular type of DSM-CC message, the dispatcher 38 is primarily responsible for directing each incoming data block to an appropriate processor. Data access system 30

currently incorporates three specific processors: a download information indication (DII) processor 44, which receives DSM-CC download information indication (DII) messages; cancel processor 46, which receives DSM-CC messages that indicate when a module has been removed from the data carousel; and data block processor 48, which receives all other types of data blocks. For the in-band QAM channel, data access system 30 also interacts with a TV manager 52, which provides the functionality to select and manipulate in-band data source.

Description Paragraph (15):
When an application requests to "open" (or retrieve) a module on the data carousel, it registers this request with the interest manager 36. The interest manager 36 maintains a list of modules that the data access system 30 is currently interested in receiving from the data carousel. Thus, an interest exists for every application request. An identifier (e.g., source id, carousel id, module id, version id, etc.) that uniquely identifies the requested module as well as an identifier for the requesting application are stored in a data structure 37 associated with the interest manager 36. Once an interest has been registered by an application 34, all data blocks that match that interest are processed by the data block processor 48.

Description Paragraph (16):
In addition to registering each application request, an open file component 36A of the interest manager 36 will interface with the BFS directory 50 to ensure that the requested module exists on the data carousel. Generally, DII messages contain a directory of all of the modules on the carousel and are periodically received (e.g., multiple times per second) by the DII processor 44. As part of this synchronization process, the open file component 36A also verifies that the module exists in accordance to the most recently received DII message. Next, it checks that the version number for the module retrieved from the BFS directory matches the version for that module included in the DII message. In this way, the interest manager 36 ensures that the applications request to open a module is synchronized with the module contained on the data carousel. Because modules are delivered on multiple sources at different rates, it is important to implement this synchronization process; otherwise an application may read data from either an older version of the module or possibly even a newer version of the module. It is also envisioned that a filter may be dynamically installed by the open file component 36A that allows DII messages for this carousel to pass through to the dispatcher 38. Once the synchronization process is complete, the filter is removed so that DII messages will not unnecessarily pass through to the dispatcher 38.

Description Paragraph (18):
Typically, a data carousel broadcasts all of the data on the carousel before it rebroadcasts any of the data. Therefore, when an application makes a new data request, the response time depends on the data's location relative to where the carousel is in the broadcast cycle. For example, a carousel having ten data blocks may be currently broadcasting the fifth data block. An application that requests data contained in the seventh block will receive that data faster than an application that requests data contained in the ninth block. If it is the eighth block that is currently being broadcast, the reverse is true; an application that requests data contained in the ninth block will receive that data faster than an application that requests data contained in the seventh block.

Description Paragraph (19):
Therefore, data block processor 48 reads the data from a data carousel as it is broadcast; it does not wait for the beginning of the data to begin reading. For instance, an application may be interested in a module contained in blocks 2 to 6. If it starts reading as block 4 is broadcast, data block processor 48 copies the data in block 4 to the application's buffer and then reads the data in each successive block that is broadcast. Subsequent data blocks associated with the requested module (i.e., blocks 5, 6, 2, and 3) are copied to the application's buffer, whereas data that is not part of the module (i.e., blocks 7, 8, 9, 10 and

1) are discarded.

Description Paragraph (21):
Decision block 106 compares the version number for each incoming data block to the expected version number contained in the BFS directory. This is important because a module can be changed on the carousel as individual data blocks for that module are being read by data access system 30. If the version numbers match, then processing proceeds to decision block 110. On the other hand, if the version number does not match, processing branches to an error subroutine 108.

Description Paragraph (22):
Next, decision block 110 determines if a data block falls within the desired range of data blocks. For each data blocks within the desired range, Block 112 sets a Read Block indicator. Data blocks outside the desired range are discarded in block 114. Read Block indicator is an array data structure used to monitor which data blocks have been read from the data carousel. Each bit in the array represents a data block in the requested module, such that if the bit is on (i.e., set to 1), then the block has been read, but if the bit is off (i.e., set to 0), then the block has not been read. Using this indicator allows data block processor 48 to receive data blocks in any order from the data carousel, and thus eliminates any unnecessary memory copies.

Description Paragraph (24):
Since the data access system 30 receives data from multiple data sources, each of which may have different content as well as different data rates, the BFS directory may not always be in synch with the content of the data carousel being broadcast to the client. For example, a module (including the BFS directory module) can be changed on the carousel as individual data blocks for that module are being read by data access system 30. Therefore, the data block processor 48 also provides a method for downloading and synchronizing the BFS directory on the client.

Description Paragraph (25):
FIG. 6 is a flowchart showing a portion of the data block processor 48 that handles the downloading of the BFS directory. Initially, when a set-top box on an (RF) broadcast network boots, a DSM-CC user-to-network configuration message is received at the client that contains the source ID, carousel ID, and module ID for the BFS directory. Based on this information, block 140 installs a filter that allows (any version of) BFS directory data blocks to pass through the dispatcher 38. As a result, the current version of the BFS directory is retrieved from the data carousel. In a preferred embodiment, BFS directory structure is only delivered over the out-of-band source (which by definition is guaranteed to be available to data access system 30), and thus this filter only need be applied to that data source.

Description Paragraph (26):
Decision block 144 determines when a BFS directory data block arrives, and, if so, block 146 reads and parses each incoming data block to construct the BFS directory structure. In the event an error occurs during the downloading process, decision block 148 returns processing to block 142 for re-installation of the initial BFS directory filter. This ensures that the current BFS directory is always downloaded from the data carousel.

Description Paragraph (27):
Without an error, block 150 checks any registered interests against the newly downloaded BFS directory. At boot up, there will be no registered interests. However, for subsequently downloaded BFS directories, DII processor 44 interfaces with the interest manager 36 to verify registered interests. If a registered interest no longer exists or has been updated on the data carousel (as indicated by the newly downloaded BFS directory), then the interest is marked for deletion. Block 152 then installs (e.g., cached in memory) the newly downloaded BFS directory onto the client for use by data access system 30.

CLAIMS:

1. A broadcast data access system including a plurality of modules, the broadcast data access system for providing non-sequential data blocks indicative of a module from a data carousel to a multimedia client, the multimedia client comprising: a plurality of requesting applications each for requesting a module; an interest manager for receiving from at least one of the plurality of requesting applications an interest indicative of the requested module and for storing the interest, wherein the interest includes a first identifier identifying the requested module and a second identifier identifying the at least one of the plurality of requesting applications; and a dispatcher for receiving the non-sequential data blocks from the data carousel, and for providing the non-sequential data blocks to the at least one of the plurality of requesting applications determined by the first and second identifiers in the interest manager, wherein the interest manager removes the interest upon completely reassembling the requested module and providing the requested module to the at least one of the plurality of requested applications.

11. The broadcast data access system of claim 1, the multimedia client further comprising a broadcast file system for including a directory of the plurality of modules on the data carousel, wherein each of the plurality of modules including an updated first identifier.

13. A broadcast data access system for providing non-sequential data blocks indicative of a module from a data carousel to a multimedia client, the multimedia client comprising: a plurality of requesting applications each for requesting a module; an interest manager for receiving from at least one of the plurality of requesting applications an interest indicative of the requested module and for storing the interest, wherein the interest includes a first identifier identifying the requested module and a second identifier for identifying the at least one of the requesting applications; and a dispatcher for receiving the non-sequential data blocks from the data carousel, for storing the non-sequential data blocks in memory, and, upon completion of the storage of all of the non-sequential data blocks associated with the requested module for providing the module to the at least one of the plurality of requesting applications, wherein the interest manager removes the interest upon the dispatcher providing the module to the at least one of the plurality of requesting applications.

17. The broadcast data access system of claim 13, the multimedia client further comprising a broadcast file system for including a directory of the plurality of modules on the data carousel, wherein each of the plurality of modules including an updated first identifier.

Previous Doc    Next Doc    Go to Doc#